

What is claimed is:

1 1. A system for efficiently forwarding client requests in a distributed
2 computing environment, comprising:
3 a socket receiving a plurality of non-proxiable requests commonly
4 addressed to an origin server from individual sending clients;
5 a time estimates generator dynamically generating, concurrent to and
6 during processing of each request, time estimates of service availability based on
7 a time-to-idle for sending the requests over each of a plurality of connections to
8 the origin server; and
9 a connection manager selecting the connection to the origin server with a
10 substantially highest service availability and a substantially lowest time-to-idle
11 and forwarding each request to the origin server using the selected connection.

1 2. A system according to Claim 1, further comprising:
2 the connection manager selecting a connection not actively sending a
3 request with a zero time-to-idle and not subject to a slow start overhead incurred
4 responsive to flow control imposed by the sending client.

1 3. A system according to Claim 2, further comprising:
2 the connection manager selecting a connection actively sending a request
3 with a time-to-idle less than the slow start overhead, plus request transfer time if
4 the connection is pipelined.

1 4. A system according to Claim 3, further comprising:
2 the connection manager selecting a connection not actively sending a
3 request with a zero time-to-idle and subject to the slow start overhead.

1 5. A system according to Claim 4, further comprising:
2 the connection manager selecting a connection actively sending a request
3 with a time-to-idle less than a connection setup overhead, plus request transfer
4 time if the connection is pipelined.

1 6. A system according to Claim 5, further comprising:

2 the connection manager selecting a new connection in the absence of an
3 existing connection with a time-to-idle less than the connection setup overhead.

1 7. A system according to Claim 5, further comprising:
2 the connection manager selecting an existing connection with the
3 substantially lowest time-to-idle.

1 8. A system according to Claim 1, wherein the distributed operating
2 environment is TCP/IP-compliant, the system further comprising:
3 the time estimates generator providing time estimates for each connection
4 comprising at least one of TCP overhead, time-to-idle, idle time, and request
5 transfer time.

1 9. A system according to Claim 8, the connection setup overhead
2 comprises TCP overhead, the system further comprising:
3 the time estimates generator calculating the TCP overhead by adding a
4 three-way handshake overhead to a slow start overhead.

1 10. A system according to Claim 8, further comprising:
2 the time estimates generator calculating the request transfer time by
3 multiplying the size of the request by an average connection speed for the origin
4 server.

1 11. A system according to Claim 8, further comprising:
2 the time estimates generator calculating the time-to-idle upon each receipt
3 of a request by adding the time-to-idle to the product of an average connection
4 speed for the origin server multiplied by the sum of the request size and an
5 estimated response size.

1 12. A system according to Claim 8, further comprising:
2 the time estimates generator calculating the time-to-idle upon writing data
3 to a socket by subtracting the time-to-idle from the product of an average
4 connection speed for the origin server multiplied by the amount of data written.

1 13. A system according to Claim 8, further comprising:
2 the time estimates generator calculating the time-to-idle upon reading data
3 from a socket, prior to header data, by subtracting the time-to-idle from the
4 product of an average connection speed for the origin server multiplied by the
5 amount of data read.

1 14. A system according to Claim 1, further comprising:
2 a proxy configured in a location comprising at least one of local to the
3 sending clients, in the infrastructure of the distributed computing environment,
4 and local to the origin server.

1 15. A method for efficiently forwarding client requests in a distributed
2 computing environment, comprising:
3 receiving a plurality of non-proxiable requests commonly addressed to an
4 origin server from individual sending clients;
5 dynamically generating, concurrent to and during processing of each
6 request, time estimates of service availability based on a time-to-idle for sending
7 the requests over each of a plurality of connections to the origin server; and
8 selecting the connection to the origin server with a substantially highest
9 service availability and a substantially lowest time-to-idle and forwarding each
10 request to the origin server using the selected connection.

1 16. A method according to Claim 15, further comprising:
2 selecting a connection not actively sending a request with a zero time-to-
3 idle and not subject to a slow start overhead incurred responsive to flow control
4 imposed by the sending client.

1 17. A method according to Claim 16, further comprising:
2 selecting a connection actively sending a request with a time-to-idle less
3 than the slow start overhead, plus request transfer time if the connection is
4 pipelined.

1 18. A method according to Claim 17, further comprising:

2 selecting a connection not actively sending a request with a zero time-to-
3 idle and subject to the slow start overhead.

1 19. A method according to Claim 18, further comprising:
2 selecting a connection actively sending a request with a time-to-idle less
3 than a connection setup overhead, plus request transfer time if the connection is
4 pipelined.

1 20. A method according to Claim 19, further comprising:
2 selecting a new connection in the absence of an existing connection with a
3 time-to-idle less than the connection setup overhead.

1 21. A method according to Claim 19, further comprising:
2 selecting an existing connection with the substantially lowest time-to-idle.

1 22. A method according to Claim 15, wherein the distributed operating
2 environment is TCP/IP-compliant, the method further comprising:
3 providing time estimates for each connection comprising at least one of
4 TCP overhead, time-to-idle, idle time, and request transfer time.

1 23. A method according to Claim 22, the connection setup overhead
2 comprises TCP overhead, the method further comprising:
3 calculating the TCP overhead by adding a three-way handshake overhead
4 to a slow start overhead.

1 24. A method according to Claim 22, further comprising:
2 calculating the request transfer time by multiplying the size of the request
3 by an average connection speed for the origin server.

1 25. A method according to Claim 22, further comprising:
2 calculating the time-to-idle upon each receipt of a request by adding the
3 time-to-idle to the product of an average connection speed for the origin server
4 multiplied by the sum of the request size and an estimated response size.

1 26. A method according to Claim 22, further comprising:

2 calculating the time-to-idle upon writing data to a socket by subtracting
3 the time-to-idle from the product of an average connection speed for the origin
4 server multiplied by the amount of data written.

1 27. A method according to Claim 22, further comprising:
2 calculating the time-to-idle upon reading data from a socket, prior to
3 header data, by subtracting the time-to-idle from the product of an average
4 connection speed for the origin server multiplied by the amount of data read.

1 28. A method according to Claim 15, further comprising:
2 providing a proxy configured in a location comprising at least one of local
3 to the sending clients, in the infrastructure of the distributed computing
4 environment, and local to the origin server.

1 29. A computer-readable storage medium holding code for performing
2 the method according to Claim 15.

1 30. A system for efficiently forwarding client requests from a proxy
2 server in a TCP/IP computing environment, comprising:
3 means for receiving a plurality of transient requests from individual
4 sending clients, each request being commonly addressed to an origin server;
5 means for dynamically calculating, concurrent to receiving and during
6 processing of each request, time estimates of TCP overhead, slow start overhead,
7 time-to-idle, and request transfer time for sending the requests over each of a
8 plurality of managed connections to the origin server;
9 means for choosing the managed connection from, in order of preferred
10 selection, a warm idle connection, an active connection with a time-to-idle less
11 than a slow start overhead, a cold idle connection, an active connection with a
12 time-to-idle less than a TCP overhead, a new managed connection, and an
13 existing managed connection with a smallest time-to-idle; and
14 means for forwarding each request to the origin server over the selected
15 managed connection.

1 31. A system according to Claim 30, further comprising:
2 means for adding the request transfer time during each active connection
3 selection if the managed connection is pipelined.

1 32. A system according to Claim 30, further comprising:
2 means for calculating the TCP overhead by adding a three-way handshake
3 overhead to a slow start overhead;
4 means for calculating the request transfer time by multiplying the size of
5 the request by an average managed connection speed for the origin server; and
6 means for calculating the time-to-idle, comprising:
7 upon each receipt of a request, means for adding the time-to-idle to
8 the product of an average managed connection speed for the origin server
9 multiplied by the sum of the request size and an estimated response size;
10 upon writing data to a socket, means for subtracting the time-to-
11 idle from the product of an average managed connection speed for the origin
12 server multiplied by the amount of data written; and
13 upon reading data from a socket, prior to header data, means for
14 subtracting the time-to-idle from the product of an average managed connection
15 speed for the origin server multiplied by the amount of data read.

1 33. A system according to Claim 30, wherein each transient request is
2 communicated in accordance with HTTP.

1 34. A method for efficiently forwarding client requests from a proxy
2 server in a TCP/IP computing environment, comprising:
3 receiving a plurality of transient requests from individual sending clients
4 into a request queue, each request being commonly addressed to an origin server;
5 dynamically calculating, concurrent to receiving and during processing of
6 each request, time estimates of TCP overhead, slow start overhead, time-to-idle,
7 and request transfer time for sending the requests over each of a plurality of
8 managed connections to the origin server;

9 choosing the managed connection from, in order of preferred selection, a
10 warm idle connection, an active connection with a time-to-idle less than a slow
11 start overhead, a cold idle connection, an active connection with a time-to-idle
12 less than a TCP overhead, a new managed connection, and an existing managed
13 connection with a smallest time-to-idle; and
14 forwarding each request to the origin server over the selected managed
15 connection.

1 35. A method according to Claim 34, further comprising:
2 adding the request transfer time during each active connection selection if
3 the managed connection is pipelined.

1 36. A method according to Claim 34, further comprising:
2 calculating the TCP overhead by adding a three-way handshake overhead
3 to a slow start overhead;
4 calculating the request transfer time by multiplying the size of the request
5 by an average managed connection speed for the origin server; and
6 calculating the time-to-idle, comprising:
7 upon each receipt of a request, adding the time-to-idle to the
8 product of an average managed connection speed for the origin server multiplied
9 by the sum of the request size and an estimated response size;
10 upon writing data to a socket, subtracting the time-to-idle from the
11 product of an average managed connection speed for the origin server multiplied
12 by the amount of data written; and
13 upon reading data from a socket, prior to header data, subtracting
14 the time-to-idle from the product of an average managed connection speed for the
15 origin server multiplied by the amount of data read.

1 37. A method according to Claim 34, wherein each transient request is
2 communicated in accordance with HTTP.

1 38. A computer-readable storage medium holding code for performing
2 the method according to Claim 34.